Fast Multipole Methods: Fundamentals & Applications

Ramani Duraiswami

Nail A. Gumerov

What is the Fast Multipole Method?

- An algorithm for achieving fast products of particular dense matrices with vectors
- Similar to the Fast Fourier Transform
  - For the FFT, matrix entries are uniformly sampled complex exponentials
- For FMM, matrix entries are
  - Derived from particular functions
  - Functions satisfy known "translation" theorems
- Name is a bit unfortunate
  - What the heck is a multipole? We will return to this …
- Why is this important?

Vectors and Matrices

d dimensional column vector **x** and its transpose

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{pmatrix} \quad \text{and} \quad \mathbf{x}^t = (x_1 \ x_2 \ \ldots \ x_d)$$

- $n \times d$ dimensional matrix **M** and its transpose $\mathbf{M}^t$

$$\mathbf{M} = \begin{pmatrix} m_{11} & m_{12} & m_{13} & \ldots & m_{1d} \\ m_{21} & m_{22} & m_{23} & \ldots & m_{2d} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ m_{n1} & m_{n2} & m_{n3} & \ldots & m_{nd} \end{pmatrix} \text{and}$$

$$\mathbf{M}^t = \begin{pmatrix} m_{11} & m_{21} & \ldots & m_{n1} \\ m_{12} & m_{22} & \ldots & m_{n2} \\ m_{13} & m_{23} & \ldots & m_{n3} \\ \vdots & \vdots & \ddots & \vdots \\ m_{1d} & m_{2d} & \ldots & m_{nd} \end{pmatrix}.$$

Matrix vector product

$$s_1 = m_{11} x_1 + m_{12} x_2 + \ldots + m_{1d} x_d$$
$$s_2 = m_{21} x_1 + m_{22} x_2 + \ldots + m_{2d} x_d$$
$$\ldots$$
$$s_n = m_{n1} x_1 + m_{n2} x_2 + \ldots + m_{nd} x_d$$

- d products and sums per line
- N lines
- Total Nd products and Nd sums to calculate N entries

- Matrix vector product is identical to a sum

$$s_i = \sum_{j=1}^{d} m_{ij} x_j$$

- So algorithm for fast matrix vector products is also a fast summation algorithm

Linear Systems

- Solve a system of equations

$$Mx=s$$

- M is a $N \times N$ matrix, x is a N vector, s is a N vector
- Direct solution (Gauss elimination, LU Decomposition, SVD, …) all need $O(N^3)$ operations
- Iterative methods typically converge in k steps with each step needing a matrix vector multiply $O(N^2)$
  - if properly designed, $k \ll N$
- A fast matrix vector multiplication algorithm ($O(N \log N)$ operations) will speed all these algorithms

Is this important?

- Argument:
  - Moore's law: Processor speed doubles every 18 months
  - If we wait long enough the computer will get fast enough and let my inefficient algorithm tackle the problem
- Is this true?
  - Yes for algorithms with same asymptotic complexity
  - No!! For algorithms with different asymptotic complexity
- For a million variables, we would need about 16 generations of Moore's law before a $O(N^2)$ algorithm was comparable with a $O(N)$ algorithm
- Similarly, clever problem formulation can also achieve large savings.

Memory complexity

- Sometimes we are not able to fit a problem in available memory
  - Don’t care how long solution takes, just if we can solve it
- To store a $N \times N$ matrix we need N^2 locations
  - 1 GB RAM = 1024^3 =1,073,741,824 bytes
  - => largest N is 32,768
- “Out of core” algorithms copy partial results to disk, and keep only necessary part of the matrix in memory
- FMM allows reduction of memory complexity as well
  - *Elements of the matrix required for the product can be generated as needed*

The need for fast algorithms

- Grand challenge problems in large numbers of variables
- Simulation of physical systems
  - Electromagnetics of complex systems
  - Stellar clusters
  - Protein folding
  - Turbulence
- Learning theory
  - Kernel methods
  - Support Vector Machines
- Graphics and Vision
  - Light scattering …

- General problems in these areas can be posed in terms of millions (10^6) or billions (10^9) of variables
- Recall Avogadro’s numer (6.022 141 99 $\times 10^{23}$ molecules/mole

Dense and Sparse matrices

- Operation estimates are for **dense matrices**.
  - Majority of elements of the matrix are ***non-zero***
- However in many applications matrices are ***sparse***
- A sparse matrix (or vector, or array) is one in which most of the elements are zero.
  - If storage space is more important than access speed, it may be preferable to store a sparse matrix as a list of (index, value) pairs.
  - For a given sparsity structure it may be possible to define a fast matrix-vector product/linear system algorithm

- Can compute

$$\begin{bmatrix} a_1 & 0 & 0 & 0 & 0 \\ 0 & a_2 & 0 & 0 & 0 \\ 0 & 0 & a_3 & 0 & 0 \\ 0 & 0 & 0 & a_4 & 0 \\ 0 & 0 & 0 & 0 & a_5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} a_1 x_1 \\ a_2 x_2 \\ a_3 x_3 \\ a_4 x_4 \\ a_5 x_5 \end{bmatrix}$$

In 5 operations instead of 25 operations

- Sparse matrices are not our concern here

Structured matrices

- Fast algorithms have been found for many dense matrices
- Typically the matrices have some *"structure"*
- Definition:
  - A dense matrix of order $N \times N$ is called structured if its entries depend on only $O(N)$ parameters.
- Most famous example – the fast Fourier transform

Fourier Matrices

A Fourier matrix of order n is defined as the following

$$F_n = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega_n & \omega_n^2 & \cdots & \omega_n^{n-1} \\ 1 & \omega_n^2 & \omega_n^4 & \cdots & \omega_n^{2(n-1)} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & \omega_n^{n-1} & \omega_n^{2(n-1)} & \cdots & \omega_n^{(n-1)(n-1)} \end{bmatrix},$$

where

$$\omega_n = e^{-\frac{2\pi i}{n}},$$

is an nth root of unity.

FFT and IFFT

The *discrete Fourier transform* of a vector x is the product $F_n x$.
The *inverse discrete Fourier transform* of a vector x is the product $F_n^* x$.

Both products can be done efficiently using the fast Fourier transform (FFT) and the inverse fast Fourier transform (IFFT) in $O(n \log n)$ time.

The FFT has revolutionized many applications by reducing the complexity by a factor of almost *n*

Can relate many other matrices to the Fourier Matrix

Circulant Matrices

$$C_n = C(x_1, ..., x_n) = \begin{bmatrix} x_1 & x_n & x_{n-1} & \cdots & x_2 \\ x_2 & x_1 & x_n & \cdots & x_3 \\ x_3 & x_2 & x_1 & \cdots & x_4 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_n & x_{n-1} & x_{n-2} & \cdots & x_1 \end{bmatrix}$$

Toeplitz Matrices

$$T_n = T(x_{-n+1}, \cdots, x_0, ..., x_{n-1}) = \begin{bmatrix} x_0 & x_1 & x_2 & \cdots & x_{n-1} \\ x_{-1} & x_0 & x_1 & \cdots & x_{n-2} \\ x_{-2} & x_{-1} & x_0 & \cdots & x_{n-3} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{-n+1} & x_{-n+2} & x_{-n+3} & \cdots & x_0 \end{bmatrix}$$

Hankel Matrices

$$H_n = H(x_{-n+1}, \cdots, x_0, \cdots, x_{n-1}) = \begin{bmatrix} x_{-n+1} & x_{-n+2} & x_{-n+3} & \cdots & x_0 \\ x_{-n+2} & x_{-n+3} & x_{-n+4} & \cdots & x_1 \\ x_{-n+3} & x_{-n+4} & x_{-n+5} & \cdots & x_2 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_0 & x_1 & x_2 & \cdots & x_{n-1} \end{bmatrix}$$

Vandermonde Matrices

$$V = V(x_0, x_1, ..., x_n) = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ x_0 & x_1 & \cdots & x_{n-1} \\ \cdots & \cdots & \cdots & \cdots \\ x_0^{n-1} & x_1^{n-1} & \cdots & x_{n-1}^{n-1} \end{bmatrix}$$

- Modern signal processing very strongly based on the FFT
- One of the defining inventions of the 20th century

Fast Multipole Methods (FMM)

- Introduced by Rokhlin & Greengard in 1987
- Called one of the 10 most significant advances in computing of the 20th century
- Speeds up matrix-vector products (sums) of a particular type

$$s(x_j) = \sum_{i=1}^{N} \alpha_i \phi(x_j - x_i), \quad \{s_j\} = [\Phi_{ji}]\{\alpha_i\}.$$

- Above sum requires $O(MN)$ operations.
- For a given precision ε the FMM achieves the evaluation in $O(M+N)$ operations.

- Can accelerate matrix vector products
  - Convert $O(N^2)$ to $O(N \log N)$
- However, can also accelerate linear system solution
  - Convert $O(N^3)$ to $O(kN \log N)$

A very simple algorithm

- Not FMM, but has some key ideas
- Consider

$$S(x_i)=\sum_{j=1}^{N} \alpha_j (x_i - y_j)^2 \qquad i=1, \ldots ,M$$

- Naïve way to evaluate the sum will require *MN* operations
- Instead can write the sum as

$$S(x_i)=(\sum_{j=1}^{N} \alpha_j)x_i^2 + (\sum_{j=1}^{N} \alpha_j y_j^2) -2x_i(\sum_{j=1}^{N} \alpha_j y_j)$$

  – Can evaluate each bracketed sum over *j* and evaluate an expression of the type

$$S(x_i)=\beta x_i^2 + \gamma -2x_i\delta$$

  – Requires O(M+N) operations
- Key idea – use of analytical manipulation of series to achieve faster summation

Approximate evaluation

- FMM introduces another key idea or "philosophy"
  - In scientific computing we almost never seek exact answers
  - At best, "exact" means to "machine precision"
- So instead of solving the problem we can solve a "nearby" problem that gives "almost" the same answer
- If this "nearby" problem is much easier to solve, and we can bound the error analytically we are done.
- In the case of the FMM
  - Manipulate series to achieve approximate evaluation
  - Use analytical expression to bound the error
- FFT is exact … FMM can be arbitrarily accurate

Some FMM algorithms

- Molecular and stellar dynamics
  - Computation of force fields and dynamics
- Interpolation with Radial Basis Functions
- Solution of acoustical scattering problems
  - Helmholtz Equation
- Electromagnetic Wave scattering
  - Maxwell's equations
- Fluid Mechanics: Potential flow, vortex flow
  - Laplace/Poisson equations
- Fast nonuniform Fourier transform

Applications – I   Interpolation

- Given a scattered data set with points and values $\{\boldsymbol{x}_i, f_i\}$
- Build a representation of the function f($\mathbf{x}$)
  - That satisfies $f(\mathbf{x}_i)=f_i$
  - Can be evaluated at new points
- One approach use "radial-basis functions"

$$f(\mathbf{x})=\sum_i^N \alpha_i R(\mathbf{x}-\mathbf{x}_i) + p(\mathbf{x})$$

$$f_{\mathrm{j}}=\sum_i^N \alpha_i R(\mathbf{x}_j-\mathbf{x}_i) + p(\mathbf{x}_j)$$

- Two problems
  - Determining α_i
  - Knowing α_i determine the product at many new points $\mathbf{x}_j$
- Both can be solved via FMM (Cherrie et al, 2001)

Applications 2

- RBF interpolation

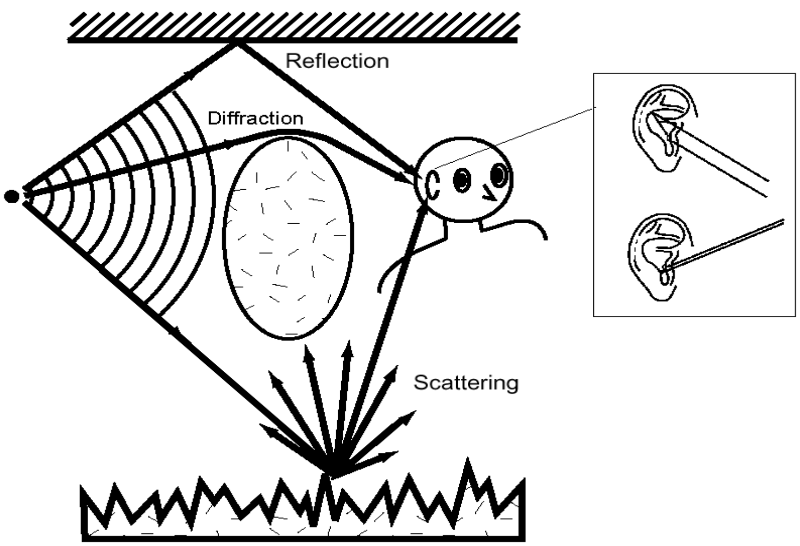
Cherrie et al 2001

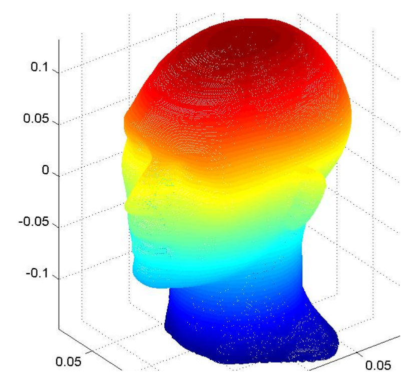
Applications 3

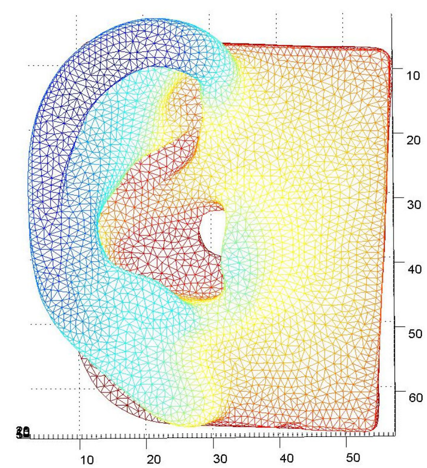
- Sound scattering off rooms and bodies
  - Need to know the scattering properties of the head and body (our interest)

$$\nabla^2 P + k^2 P = 0 \qquad \frac{\partial P}{\partial n} + i\sigma P = g \qquad \lim_{r\to\infty} r\left(\frac{\partial P}{\partial r} - ikP\right) = 0$$

$$C(x)p(x) = \int_{\Gamma_y}\left[G(x,y;k)\frac{\partial\ p(y)}{\partial\ n_y} - \frac{\partial\ G(x,y;k)}{\partial\ n_y}p(y)\right]d\Gamma_y \qquad G(\mathbf{x},\mathbf{y}) = \frac{e^{ik|\mathbf{x}-\mathbf{y}|}}{4\pi|\mathbf{x}-\mathbf{y}|}$$

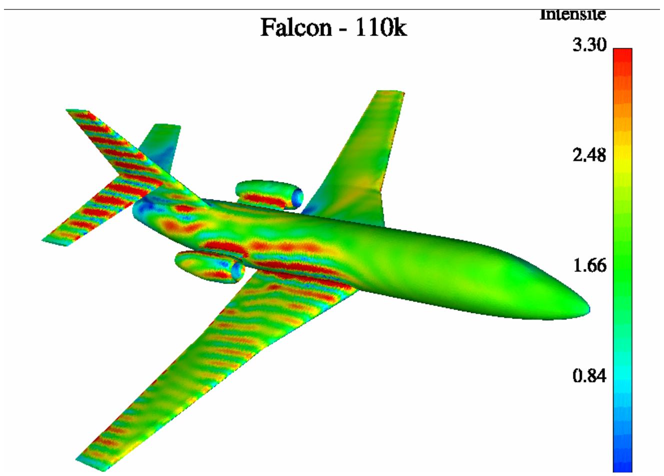






EM wave scattering

- Similar to acoustic scattering
- Send waves and measure scattered waves
- Attempt to figure out object from the measured waves
- Need to know "Radar cross-section"
- Many applications
  - Light scattering
  - Radar
  - Antenna design
  - ….



Darve 2001

Molecular and stellar dynamics

- Many particles distributed in space
- Particles exert a force on each other
- Simplest case force obeys an inverse-square law (gravity, coulombic interaction)

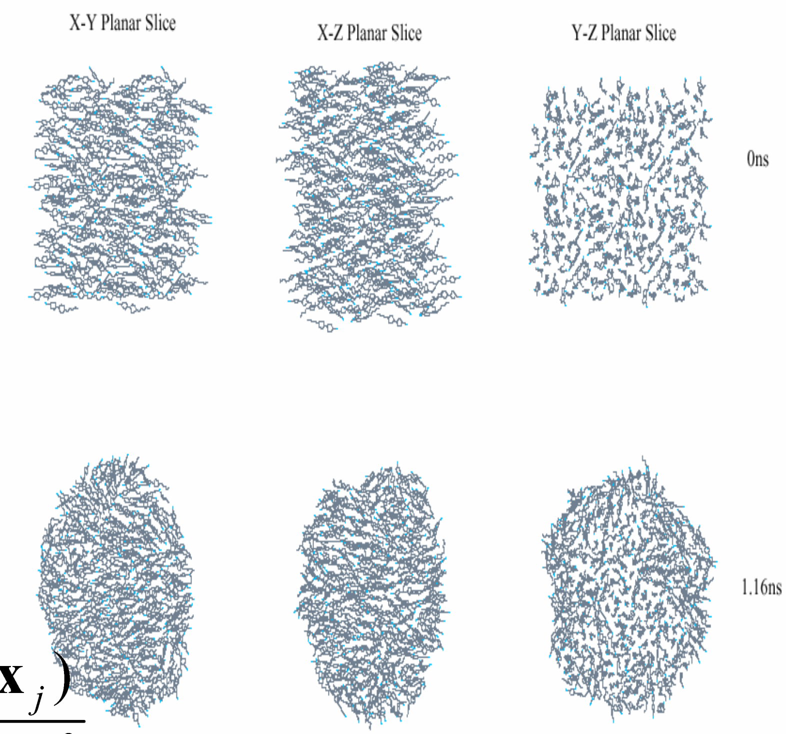


Figure 10: Slice views of the 5CB cluster at time 0 and 1.16 ns. The slices are passing the spheric center with thickness of 20 Å

$$\frac{d^2\mathbf{x}_i}{dt^2} = F_i, \qquad F_i = \sum_{\substack{j=1 \\ j\neq i}}^{N} q_i q_j \frac{(\mathbf{x}_i - \mathbf{x}_j)}{|\mathbf{x}_i - \mathbf{x}_j|^3}$$

Fluid mechanics

- Incompressible Navier Stokes Equation

$$\nabla \cdot \mathbf{u} = 0$$

$$\rho\left(\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u}\right) + \nabla p = \mu \nabla^2 \mathbf{u}$$

$$\mathbf{u} = \nabla \phi + \nabla \times \mathbf{A}$$

- Laplace equation for potential and Poisson equation for vorticity
- Solved via particle methods …

Asymptotic Equivalence

- $f(n) \sim g(n)$

$$\lim_{n\to\infty}\left(\frac{f(n)}{g(n)}\right) = 1$$

Little Oh

- *Asymptotically smaller*:

  - $f(n) = o(g(n))$

$$\lim_{n \to \infty} \left(\frac{f(n)}{g(n)} \right) = 0$$

Big Oh

- *Asymptotic Order of Growth:*
  - $f(n) = O(g(n))$

$$\limsup_{n \to \infty} \left(\frac{f(n)}{g(n)} \right) < \infty$$

The Oh's

If $f = \mathrm{o}(g)$ or $f \sim g$ then $f = \mathrm{O}(g)$

$\lim = 0$ $\quad \lim = 1$ $\quad \lim < \infty$

The Oh's

If $f = \mathrm{o}(g)$, then $g \neq \mathrm{O}(f)$

$$\lim \frac{f}{g} = 0 \qquad \lim \frac{g}{f} = \infty$$

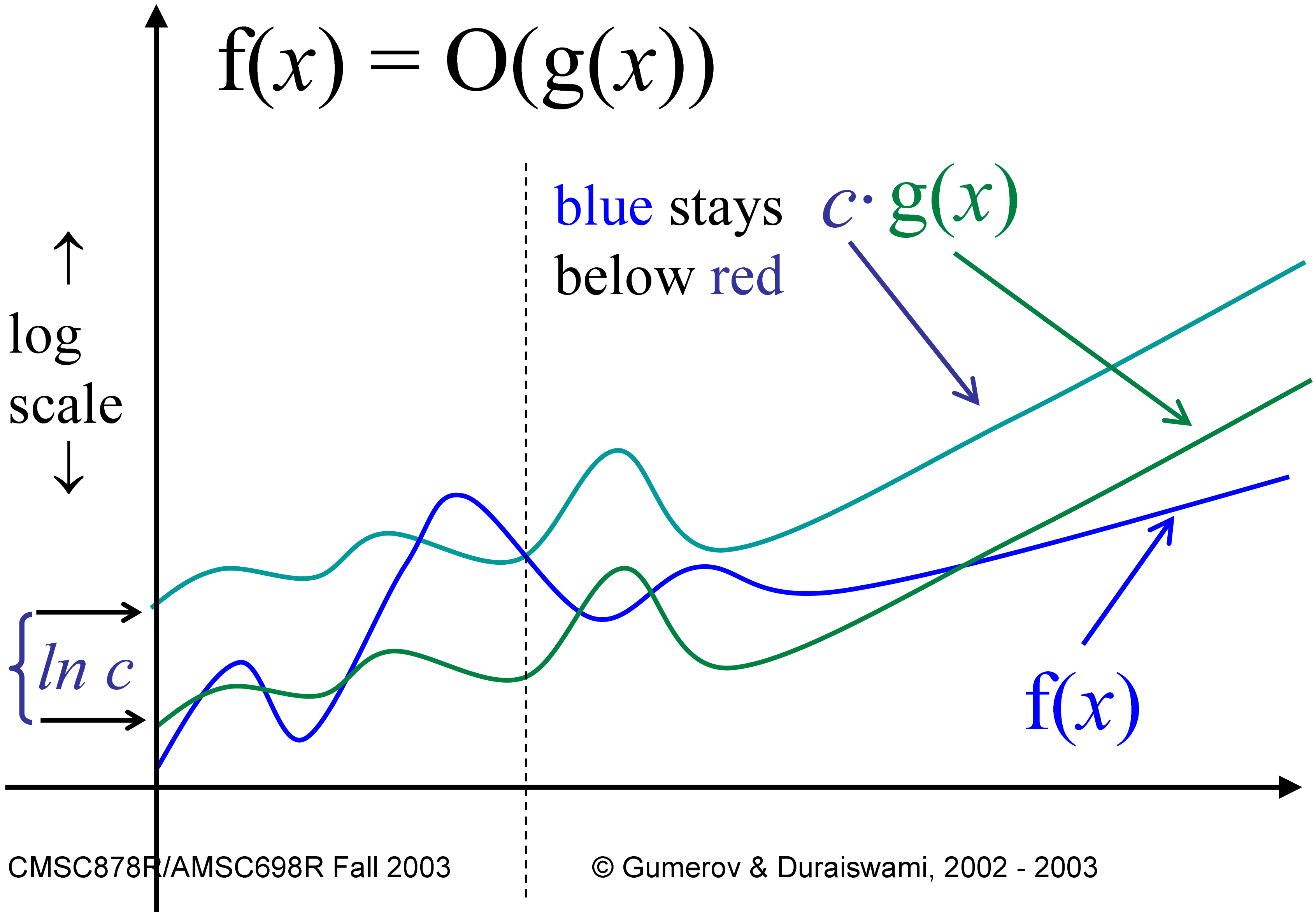
Big Oh

- Equivalently,

$$\bullet \mathrm{f}(n) = \mathrm{O}(\mathrm{g}(n))$$

$$\exists \mathrm{c}, n_0 \geq 0 \ \forall n \geq n_0 \ \ |\mathrm{f}(n)| \leq \mathrm{c} \cdot \mathrm{g}(n)$$

Big Oh



Complexity

- The most common complexities are
  - O(1) - not proportional to any variable number, i.e. a fixed/constant amount of time
  - O(N) - proportional to the size of N (this includes a loop to N and loops to constant multiples of N such as 0.5N, 2N, 2000N - no matter what that is, if you double N you expect (on average) the program to take twice as long)
  - O(N^2) - proportional to N squared (you double N, you expect it to take four times longer - usually two nested loops both dependent on N).
  - O(log N) - this is tricker to show - usually the result of binary splitting.
  - O(N log N) this is usually caused by doing log N splits but also doing N amount of work at each "layer" of splitting.

Theta

Same Order of Growth:

- $f(n) = \Theta(g(n))$

$$f(n) = O(g(n)) \text{ and } g(n) = O(f(n))$$

Log complexity

- If you half data at each stage then number of stages until you have a single item is given (roughly) by $\log_2$ N. => binary search takes $\log_2$ N time to find an item.
- All logs grow a constant amount apart (homework)
  - So we normally just say log N not $\log_2$ N.
- Log N grows very slowly

History of FMM

- Rokhlin and Greengard
- Greengard, ACM thesis award
- Rokhlin & Greengard Steele prize
- Regular FMM
- Complexity
- Translation
- Chew, Darve, Michielssen

Brief Historical Review on Fast Multipole Methods

Outline

- Separable (Degenerate) Kernels
- Problems with Infinite Series
- First Fast Solvers
- 2D Laplace Equation
- 3D Laplace Equation
- 2D Poisson Equation
- Fast Gauss Transform
- 2D Helmholtz Equation
- 3D Helmholtz Equation
- 3D Maxwell Equations
- 1D Problems
- Other Equations

Separable (Degenerate) Kernels

Compute matrix-vector product

$$\mathbf{v} = \mathbf{A}\mathbf{u},$$

or sums

$$v_j = \sum_{i=1}^{N} u_i A(x_i, y_j), \quad j = 1, \ldots, M.$$

Fast computation in case of degenerate (separable) kernel:

$$A(x_i, y_j) = \sum_{m=1}^{n} \varphi_m(x_i)\psi_m(y_j)$$

$$v_j = \sum_{i=1}^{N} u_i \sum_{m=1}^{n} \varphi_m(x_i)\psi_m(y_j) = \sum_{m=1}^{n} \psi_m(y_j) \sum_{i=1}^{N} u_i \varphi_m(x_i) = \sum_{m=1}^{n} c_m \psi_m(y_j),$$

where

$$c_m = \sum_{i=1}^{N} u_i \varphi_m(x_i).$$

Authors: Unknown.

Problems with Infinite Series

The case of degenerate kernels is not the FMM!

Compute matrix-vector product

$$\mathbf{v} = \mathbf{A}\mathbf{u},$$

or sums

$$v_j = \sum_{i=1}^{N} u_i A(x_i, y_j), \quad j = 1, \ldots, M.$$

Non-degenerate kernel:

$$A(x_i, y_j) = \sum_{m=1}^{\infty} \varphi_m(x_i)\psi_m(y_j) = \sum_{m=1}^{p} \varphi_m(x_i)\psi_m(y_j) + Error(p; x_i, y_j)$$

where p is the truncation number.

Features of the FMM:

1). Factorization is not obvious and should be selected somehow.
2). Error bounds should be established.
3). Series converge in some spatial domains. Need to have data structures and translation technique to avoid divergent series and uncontrolled error.
4) A lot of analytical work!

First Fast Solvers

Fast computation of the Laplacian gravitational fields for interstellar interactions:

A.W. Appel (1985) An efficient program for many-body simulation, *SIAM J. Stat. Comp.,* vol. 6, no. 1, 85-103.

J. Barnes & P. Hut (1986) A hierarchical O($N\log N$) force calculation algorithm, *Nature*, 234, 446-449.

2D Laplace Equation

$$\nabla^2\Phi = \frac{\partial^2\Phi}{\partial x^2} + \frac{\partial^2\Phi}{\partial y^2} = 0.$$

Fundamental solution (charge, monopole, source, free field Green's function):

$$G_{x_0,y_0}(x,y) = -\frac{1}{2\pi}\ln r, \quad r = \sqrt{(x-x_0)^2 + (y-y_0)^2}.$$

Satisfies

$$\frac{\partial^2 G_{x_0,y_0}}{\partial x^2} + \frac{\partial^2 G_{x_0,y_0}}{\partial y^2} = \delta(\mathbf{x}-\mathbf{x}_0), \quad \mathbf{x} = (x,y), \quad \mathbf{x}_0 = (x_0,y_0).$$

Field generated by a set of N monopoles:

$$\Phi(\mathbf{x}) = \sum_{i=1}^{N} q_i G_{\mathbf{x}_i}(\mathbf{x}) = \sum_{i=1}^{N} q_i G(\mathbf{x}-\mathbf{x}_i).$$

L. Greengard and V. Rokhlin, "A Fast Algorithm for Particle Simulations," J. Comput. Phys., 73, December 1987, pages 325348.135, 280-292 (1997).

L. Greengard, The rapid evaluation of potential fields in particle systems. MIT Press, Cambridge, 1988.

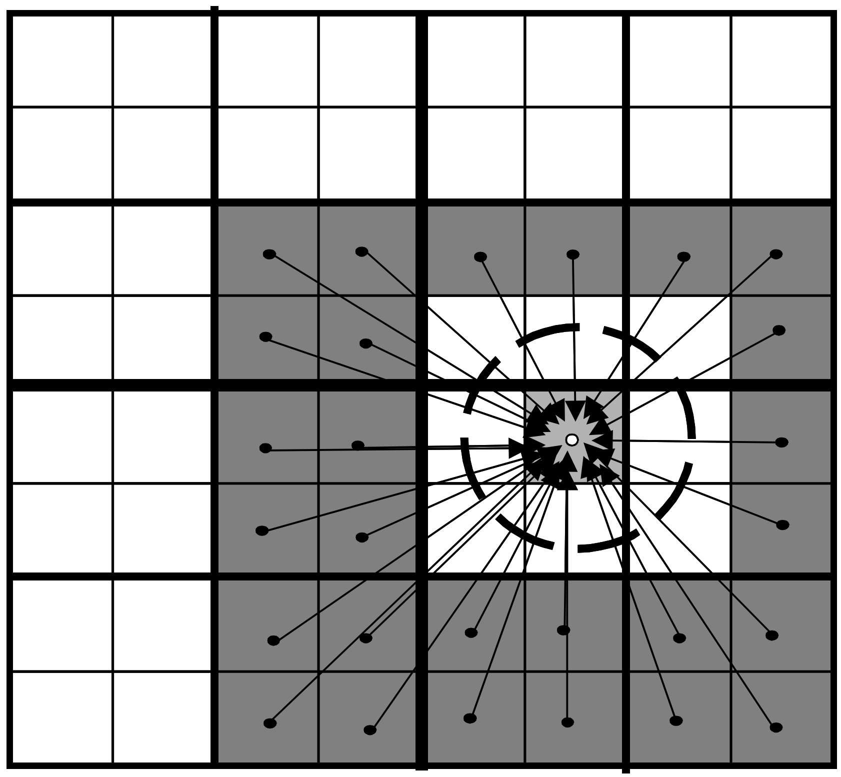
1). Introduced translation operators for 2D Laplace Equation;

2). Introduced hierarchical space subdivision based on quad-trees for data structuring in the FMM.

3). First known publications on the FMM.

Also known as MLFMA (MultiLevel Fast Multipole Algorithm)

2D Laplace Equation (Greengard's scheme of translation)



3D Laplace Equation

$$\nabla^2\Phi = \frac{\partial^2\Phi}{\partial x^2} + \frac{\partial^2\Phi}{\partial y^2} + \frac{\partial^2\Phi}{\partial z^2} = 0.$$

Fundamental solution (charge, monopole, source, free field Green's function):

$$G_{x_0,y_0,z_0}(x,y,z) = \frac{1}{4\pi r}, \quad r = \sqrt{(x-x_0)^2 + (y-y_0)^2 + (z-z_0)^2}.$$

Satisfies

$$\frac{\partial^2 G_{x_0,y_0,z_0}}{\partial x^2} + \frac{\partial^2 G_{x_0,y_0,z_0}}{\partial y^2} + \frac{\partial^2 G_{x_0,y_0,z_0}}{\partial z^2} = -\delta(\mathbf{x}-\mathbf{x}_0), \quad \mathbf{x} = (x,y,z), \quad \mathbf{x}_0 = (x_0,y_0,z_0).$$

Field generated by a set of N monopoles:

$$\Phi(\mathbf{x}) = \sum_{i=1}^{N} q_i G_{\mathbf{x}_i}(\mathbf{x}) = \sum_{i=1}^{N} q_i G(\mathbf{x}-\mathbf{x}_i).$$

L. Greengard & V. Rokhlin, Rapid evaluation of potential fields in three dimensions. Vortex Methods, C. Anderson & C. Greengard (eds.). Lecture Notes in Mathematics, vol. 1360, Springer-Verlag, 1988.

L. Greengard, The rapid evaluation of potential fields in particle systems. MIT Press, Cambridge, 1988.

1). Introduced translation operators for 3D Laplace Equation;

2). Introduced hierarchical space subdivision based on oct-trees for data structuring in the FMM.

One of the latest developments:

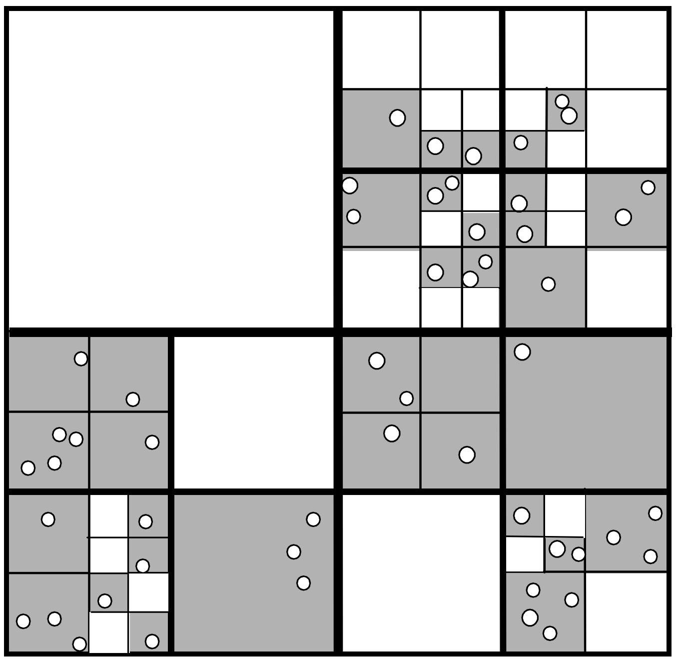
H. Cheng, L. Greengard & V. Rokhlin (1999) A fast adaptive multipole algorithm in three dimensions. J. Comp. Physics, 155, 468-498.

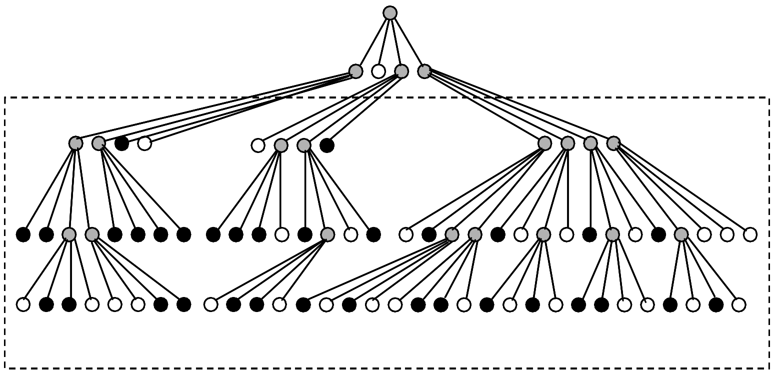
Adaptive FMM for 2D Poisson Equation

$$\nabla^2 \Phi = \frac{\partial^2 \Phi}{\partial x^2} + \frac{\partial^2 \Phi}{\partial y^2} = F(x,y).$$

L. Van Dommelen & E.A. Rundensteiner, Fast, adaptive summation of point sources in the two-dimensional poisson equation. J. Comp. Physics, 83, 126-147, 1989.

1). Introduced an adaptive quad-tree space subdivision for 2D Poisson equation. Good for very non-uniform source point distributions.





Fast Gauss Transform

$$\Phi(\mathbf{x}) = \sum_{i=1}^{N} q_i e^{-|\mathbf{x}-\mathbf{x}_i|^2/\delta}$$

1). Use of the Hermit expansions and Taylor series for different domains (far field and near field).
2). Spatial grouping based on source and evaluation points location using interaction lists.

L. Greengard & J. Strain (1991) The Fast Gauss Transform, SIAM J. Stat. Comp., 12, 1, 79-94.

J. Strain. The fast gauss transform with variable scales. SIAM J. Sci. Comput., vol. 12, pp. 1131--1139, 1991.

2D Helmholtz Equation

$$\nabla^2\Phi + k^2\Phi = \frac{\partial^2\Phi}{\partial x^2} + \frac{\partial^2\Phi}{\partial y^2} + k^2\Phi = 0.$$

Fundamental solution (charge, monopole, source, free field Green's function):

$$G_{x_0,y_0}(x,y) = \frac{1}{2\pi}H_0^{(1)}(kr), \quad r = \sqrt{(x-x_0)^2+(y-y_0)^2}.$$

$H_0^{(1)}(kr)$ is the first kind Hakel function.

Satisfies

$$\frac{\partial^2 G_{x_0,y_0}}{\partial x^2} + \frac{\partial^2 G_{x_0,y_0}}{\partial y^2} + k^2\Phi = -\delta(\mathbf{x}-\mathbf{x}_0), \quad \mathbf{x} = (x,y), \quad \mathbf{x}_0 = (x_0,y_0).$$

Field generated by a set of N monopoles:

$$\Phi(\mathbf{x}) = \sum_{i=1}^{N} q_i G_{\mathbf{x}_i}(\mathbf{x}) = \sum_{i=1}^{N} q_i G(\mathbf{x}-\mathbf{x}_i).$$

V. **Rokhlin (1990) Rapid solution of integral equations of scattering theory in two dimensions**.

1). Translation operators for 2D Helmholtz Equation;
2). Error bounds;
2). Spatial grouping.

3D Helmholtz Equation

$$\nabla^2\Phi + k^2\Phi = \frac{\partial^2\Phi}{\partial x^2} + \frac{\partial^2\Phi}{\partial y^2} + \frac{\partial^2\Phi}{\partial z^2} + k^2\Phi = 0.$$

Fundamental solution (charge, monopole, source, free field Green's function):

$$G_{x_0,y_0}(x,y) = \frac{1}{4\pi r}e^{ikr}, \quad r = \sqrt{(x-x_0)^2+(y-y_0)^2+(z-z_0)^2}.$$

Satisfies

$$\frac{\partial^2 G_{x_0,y_0}}{\partial x^2} + \frac{\partial^2 G_{x_0,y_0}}{\partial y^2} + \frac{\partial^2 G_{x_0,y_0,z_0}}{\partial z^2} + k^2\Phi = -\delta(\mathbf{x}-\mathbf{x}_0), \quad \mathbf{x} = (x,y), \quad \mathbf{x}_0 = (x_0,y_0).$$

Field generated by a set of N monopoles:

$$\Phi(\mathbf{x}) = \sum_{i=1}^{N} q_i G_{\mathbf{x}_i}(\mathbf{x}) = \sum_{i=1}^{N} q_i G(\mathbf{x}-\mathbf{x}_i).$$

R. Coifman, V. Rokhlin, and S. Wandzura (1993) The fast multipole method for the wave equation: A pedestrian prescription. IEEE Ant. Propag. Mag., vol. 35, no. 3, 7-12.

1). Translation operators for 3D Helmholtz Equation;

2). Spatial grouping.

3D Maxwell Equations

$$\nabla \times \mathbf{E} = -\mu \frac{\partial \mathbf{H}}{\partial t},$$

$$\nabla \times \mathbf{H} = \epsilon \frac{\partial \mathbf{E}}{\partial t},$$

$$\nabla \cdot \mathbf{E} = 0,$$

$$\nabla \cdot \mathbf{H} = 0,$$

\# (1)

where $\mathbf{E}$ and $\mathbf{H}$ are the electric and magnetic field vectors, and μ and ϵ are permeability and permittivity in the medium, respectively. In the case of vacuum we have

$$\mu = \mu_0, \quad \epsilon = \epsilon_0, \quad c = (\mu_0 \epsilon_0)^{-1/2},$$

where c is the speed of light in a vacuum, $c \approx 3 \cdot 10^8$ m/s.

C.C. Lu & W.C. Chew (1994) A multilevel algorithm for solving boundary-value scattering. Micro. *Opt. Tech. Lett.*, vol. 7, no. 10, 466-470.

J.M. Song & W.C. Chew (1995) Multilevel fast-multipole algorithm for solving combined field integral equations of electromagnetic scattering. *Micro. Opt. Tech*. Lett., vol. 10, no. 1, 14-19.

B. Dembart & E. Yip (1995) A 3D fast multipole method for electromagnetics with multiple levels. Ann. Rev. Prog. Appl. Computat. Electromag., vol. 1, 621-628.

1D Problems: Interpolation, Differentiation, Integration

Fast Algorithms for Polynomial Interpolation, Integration, and Differentiation
A. Dutt, M. Gu, V. Rokhlin

SIAM Journal on Numerical Analysis, Vol. 33, No. 5. (Oct., 1996), pp. 1689-1711.

1). Considered the FMM for fast Lagrange polynomial interpolation;
2). Fast summation and operations with series of polynomials.

Other Equations

- Biharmonic (Stokes Flows)
- Yukawa Potentials (molecular dynamics)
- RBF (J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, T. R. Evans, ``Reconstruction and Representation of 3D Objects with Radial Basis Functions," Proc. ACM Siggraph pp. 67-76, August 2001.)

Our papers …

